Application of the Newton–Krylov Method to Geophysical Flows

JON REISNER, VINCENT MOUSSEAU, AND DANA KNOLL

Los Alamos National Laboratory, Los Alamos, New Mexico

(Manuscript received 2 May 2000, in final form 16 February 2001)

ABSTRACT

An implicit nonlinear algorithm, the Newton–Krylov method, for the efficient and accurate simulation of the Navier–Stokes equations, is presented. This method is a combination of a nonlinear outer Newton-based iteration and a linear inner conjugate residual (Krylov) iteration but does not require the explicit formation of the Jacobian matrix. This is referred to here as Jacobian-free Newton–Krylov (JFNK). The mechanics of the method are quite simple and the method has been previously used to solve a variety of complex coupled nonlinear equations. Like most Krylov-based schemes, the key to the efficiency of the method is preconditioning. Details concerning how preconditioning is implemented into this algorithm will be illustrated in a simple one-dimensional shallow-water framework. Another important aspect of this work is examining the accuracy and efficiency of the Newton–Krylov method against an explicit method of averaging (MOA) approach. This will aid in the determination of regimes for which implicit techniques are accurate and/or efficient. Finally, results from the Navier–Stokes fluid solver used in this paper are presented. This solver employs both the JFNK and MOA approaches, and it is reasonably efficient and accurate over a large parameter space. As an illustration of the robustness of this fluid solver two different flow regimes will be shown: two-dimensional hydrostatic mountain-wave flow employing a broad mountain and two-dimensional nonhydrostatic flow employing a steep mountain and high spatial resolution.

1. Introduction

Atmospheric motions are described by coupled nonlinear partial differential equations. The equations not only describe the dynamical processes such as fluid motion, but also the physical processes such as atmospheric radiation, chemistry, etc. Because few analytical solutions exist, in order for solutions to be obtained, the equations are usually discretized and solved numerically. How the equations are discretized is a matter of choice and may involve either implicit and/or explicit temporal differencing; however, both implicit and explicit approaches typically attempt to take advantage of the disparity of timescales present in the atmosphere between those of material motions and those of the fastest waves (e.g., sound waves). For example, explicit approaches usually employ a technique called splitting (Skamarock and Klemp 1992) in which the fastest waves of the system are treated in a computationally efficient manner by splitting them off from the rest of the dynamical equations. In contrast, implicit approaches attempt to solve linear Poisson/Helmholtz equations in which the fastest waves are either eliminated or damped by the implicit solver (Clark 1977; Smolarkiewicz and Margolin 1994; Skamarock et al. 1997).

As discussed in Reisner et al. (2000), problems may arise when using explicit splitting approaches in a compressible framework, primarily that the monotonicity of the advected variables is difficult to preserve. In that paper, the use of a little-known technique, the method of averaging (MOA), was discussed and was found to allow advected variables to remain monotone, even when extreme atmospheric events such as occur in wildfires are simulated. The MOA is a nested algorithm in which advective velocities and forces are computed in a inner loop using first-order numerics, and then used in a more costly outer loop that employs a higher-order forward-in-time advective scheme. As mentioned in that paper, a potential problem does exist with the MOA approach. Namely, as the disparity of scales increases, the approach becomes less computationally efficient. To circumvent this problem we have investigated using a robust nonlinear implicit solver, the Jacobian-free Newton–Krylov (JFNK) method (Knoll et al. 1996; Knoll et al. 1999a), to implicitly solve the nonlinear Navier–Stokes equations. Unique aspects of the JFNK approach are that it enables researchers to easily investigate the importance of including nonlinearities in the discretized equations and does not require that an elliptic-like Poisson/Helmholtz equation be solved. For example, the pressure gradient term in the Navier–Stokes equations

Corresponding author address: Dr. Jon M. Reisner, Los Alamos National Laboratory, Atmospheric and Climate Science, EES-8, MS D401, Los Alamos, NM 87545.
E-mail: reisner@lanl.gov

© 2001 American Meteorological Society

involves the product of density and pressure and in almost all atmospheric models the discretized representation of this expression is linearized. We will show during the course of this paper how this term, or potentially any other nonlinear term, can be systematically modeled using the JFNK method. Since compressible models allow for the use of either explicit or implicit numerical formulations, another important aspect of this work will be to highlight regimes in which the JFNK solver becomes both numerically more efficient and accurate than the MOA approach.

The efficiency of the JFNK method is crucially dependent on the ability of the preconditioner to reduce the number of iterations required by the Krylov solver. Typically, considerable flexibility exists both in the choice of a preconditioner (Saad 1995; Skamarock et al. 1997) and also on the level of approximations made in the preconditioning matrix. If physical effects such as turbulence, radiation, or latent heat release are included in the JFNK method, the preconditioner can become complicated, and in certain circumstances the majority of the computational time is used within the preconditioner and not in the Krylov solver. Hence, constructing an efficient preconditioner for a Krylov solver requires knowledge of the problem physics and some trial and error. However, successful construction of an efficient preconditioner can result in a significant increase in computational efficiency. In section 2 we present the basics of JFNK. In section 3 we will detail how a simple preconditioner is incorporated into the JFNK method as well as illustrate the details of the JFNK approach on the one-dimensional shallow-water equations, and in this section we will show results from the shallow-water model. In the fourth section, results from a fully compressible model employing the JFNK solver will be shown. In the final section we offer some concluding remarks.

2. Jacobian-free Newton–Krylov method

In this section we will describe the basics of the JFNK method, and the preconditioning of this method, in a generic setting. The following section will be more specific, by presenting the application of JFNK to the shallow-water equations. Significant contributions have been made to the ideas behind JFNK from both the partial differential equations (PDEs) community and the optimization community. We will provide reference to both communities, but will focus on the PDE community since it is the work we are most familiar with.

a. Basics

Newton's method requires the solution of the linear system,

$$\mathbf{J}^{nk}\boldsymbol{\delta}\mathbf{u}^{nk} = -\mathcal{F}(\mathbf{u}^{nk}), \qquad \mathbf{u}^{nk+1} = \mathbf{u}^{nk} + d\boldsymbol{\delta}\mathbf{u}^{nk}, \quad (1)$$

where $\mathbf{J}$ is the Jacobian matrix, $\mathcal{F}(\mathbf{u})$ is the nonlinear system of equations (the discretized partial differential equations), $\mathbf{u}$ is the state vector, $\boldsymbol{\delta}\mathbf{u}$ is the Newton update vector, d is an adaptively chosen damping scalar, and nk is the Newton iteration level (the superscript n will be used later for time level). Note, in this study $d = 1$. Equation (1) is iterated until $\|\mathcal{F}(\mathbf{u}^{nk})\|_2 < \epsilon_{nk}$ with ϵ_{nk} a specified nonlinear tolerance. In the next section we will briefly discuss how the accuracy of a particular solution depends on ϵ_{nk}.

For our Krylov solver, we use the GCR(k) method of Eisenstat et al. (1983), which is a nonsymmetric variant of the conjugate gradient method. This solver requires that an initial residual be specified. Suppressing the nonlinear (Newton) iteration index, nk, an initial linear residual, $\mathbf{r}^o$, can be expressed as follows:

$$\mathbf{r}^o = -\mathcal{F}(\mathbf{u}) - \mathbf{J}\boldsymbol{\delta}\mathbf{u}^o,$$

with $\boldsymbol{\delta}\mathbf{u}^o$ usually being set to zero.

Along with the initial residual, to approximately invert (1) for $\boldsymbol{\delta}\mathbf{u}$ each iteration, l, of our chosen Krylov algorithm requires the action of the Jacobian in the form of matrix-vector products, which may be approximated by a first-order Taylor series expansion (Chan and Jackson 1984; Brown and Saad 1990):

$$\mathbf{J}\mathbf{r}^l \approx [\mathcal{F}(\mathbf{u} + \epsilon\mathbf{r}^l) - \mathcal{F}(\mathbf{u})]/\epsilon, \quad (2)$$

$\mathbf{r}^l$ being a residual vector computed by our chosen Krylov solver and ϵ is a small perturbation. This is not a finite-difference approximation to the Jacobian, it is a finite-difference approximation to the Jacobian matrix times a vector. For a simple derivation of this approximation see Knoll et al. (1999a). To evaluate the small perturbation, ϵ, we use

$$\epsilon = \frac{1}{N_{\text{tot}}\|\mathbf{u}\|_2}\sum_{m=1}^{N_{\text{tot}}}(a|u_m|),$$

where N_{tot} is the total linear system dimension and a is a constant whose magnitude is approximately the square root of machine roundoff ($a = 10^{-8}$ in this study). Other options for choosing ϵ are discussed in Brown and Saad (1990).

Since the use of an iterative technique to solve (1) does not require the exact solution of the linear system, the resulting algorithm is categorized as an ‘‘inexact’’ Newton's method within the PDE community (Dembo et al. 1982; Eisenstat and Walker 1996), and as a ‘‘truncated’’ Newton method within the optimization community (Dembo and Steilhaug 1983). We employ the following inexact convergence criteria on the linear iteration:

$$\|\mathbf{J}^{nk}\boldsymbol{\delta}\mathbf{u}^{nk} + \mathcal{F}(\mathbf{u}^{nk})\|_2 < \gamma\, \|\mathcal{F}(\mathbf{u}^{nk})\|_2, \quad (3)$$

where γ is a constant less than one. Keeping γ small, which is required for Newton-like nonlinear convergence, can put a significant strain on the Krylov solver, especially as the dimension of the linear system grows. Hence, there is a trade-off between the effort required to solve the linear system to a tight tolerance and the resulting required number of nonlinear iterations. A relatively large value for γ will result in less work for the Krylov method but more nonlinear iterations, whereas a small value for γ will result in more Krylov iterations per Newton iteration with fewer Newton iterations. Examples of this trade-off between total nonlinear iterations and CPU time are given in McHugh and Knoll (1994) and Pernice and Walker (1998). Only constant values of γ are considered in this study, and these values will be such that true quadratic convergence of Newton's method is not observed. For further discussion on the subject of choosing γ see Eisenstat and Walker (1996) or Nash and Sofer (1990) and for the effect of different choices of γ on the solution of the Navier–Stokes equations see McHugh and Knoll (1994) and Pernice and Walker (1998).

b. *Preconditioning the JFNK method*

The purpose of preconditioning is to efficiently cluster eigenvalues of the iteration matrix, which will in turn reduce the required number of Krylov iterations. Traditionally, for linear problems, one chooses a few iterations of a simple iterative method (applied to the system matrix) as a preconditioner. Here, we will consider preconditioning that does not form the system matrix, $\mathbf{J}$, as well as preconditioning that does. The motive for calling this method Jacobian free and not matrix free is that a matrix is formed for the preconditioner. Preconditioning methods that do not form all the elements of the Jacobian have been simply called matrix free (Chan and Jackson 1984; Nash 1985; Qin et al. 2000) in the literature. Additionally, we believe the primary benefits of the ''matrix-free'' approximation to be when the vector, $\mathbf{u}$, contains more than one variable and hence forming the primary elements of the Jacobian could exceed the memory of some computers.

In principle one can use left or right preconditioning. We have chosen left preconditioning here. Using left preconditioning within our chosen Krylov solver requires the following two steps: 1) precondition the initial residual vector (iteratively, and not to convergence) $\mathbf{p}^o = \mathbf{P}^{-1}\mathbf{r}^o$ with the vector $\mathbf{p}^o$ being subsequently used in the Krylov solver and $\mathbf{P}^{-1}$ the inverse to the preconditioning matrix, and 2) precondition subsequent residual vectors, $\mathbf{q}^l = \mathbf{P}^{-1}\mathbf{r}^l$, with $\mathbf{q}^l$ being used to compute $\mathbf{J}\mathbf{q}^l$.

This procedure closely follows that which was described by Skamarock et al. (1997), except for the approximation to the matrix-vector multiplication required to compute $\mathbf{J}\mathbf{q}^l$. Only the matrix (or matrix elements) that is (are) required for $\mathbf{P}^{-1}$ need be formed and stored. There are two choices to be made: 1) What linearization should be used to form the matrix elements required in $\mathbf{P}^{-1}$? 2) What linear iterative method should be used for $\mathbf{q}^l = \mathbf{P}^{-1}\mathbf{r}^l$?

Examples of various preconditioning strategies applied to date within the JFNK framework include the following:

1) using a low-order spatial discretization to precondition a higher-order discretization (Knoll 1998),
2) using a simple Picard linearization to form the preconditioner (Knoll et al. 1999b),
3) ignoring specific physics terms when forming the preconditioner (Brown and Saad 1990; Knoll et al. 1999b), and
4) using a time splitting method as a preconditioner (Mousseau et al. 2000).

As an example of a simple preconditioner that requires the formation of only part of $\mathbf{P}$ consider a multipass Jacobi iteration for $\mathbf{P}\mathbf{q} = \mathbf{r}$ with $\mathbf{P} = \mathbf{L} + \mathbf{D} + \mathbf{U}$, $\mathbf{D}$ being the main diagonal, and $\mathbf{L}(\mathbf{U})$ being the lower (upper) part of the preconditioner matrix. This can be written as

$$\mathbf{q}^{k+1} = \mathbf{D}^{-1}[\mathbf{r} - (\mathbf{L} + \mathbf{U})\mathbf{q}^k],$$

where superscript k is the Jacobi iteration level. This requires the formation of $\mathbf{D}$, and $\mathbf{L}$, and $\mathbf{U}$, that is, $\mathbf{P}$. A reduced storage approach exploits the relation, $(\mathbf{L} + \mathbf{U})\mathbf{q}^k = \mathbf{P}\mathbf{q}^k - \mathbf{D}\mathbf{q}^k$, and iterates the equation,

$$\mathbf{q}^{k+1} = \mathbf{D}^{-1}[\mathbf{r} - (\mathbf{P}\mathbf{q}^k - \mathbf{D}\mathbf{q}^k)],$$

where $\mathbf{P}\mathbf{q}^k$ is formed in a matrix-free fashion. Thus one only needs to form $\mathbf{D}$ (a vector for a scalar problem). It is straightforward to implement a symmetric successive over relaxation (SSOR) in a similar reduced storage manner (Chan and Jackson 1984), as well and an alternate direction implicit (ADI) approach (Qin et al. 2000).

3. Application of the JFNK method to the shallow-water problem

a. *Shallow-water model*

For our demonstration of the JFNK approach we have chosen a simple model problem that solves the one-dimensional shallow-water equations in flux form with no bottom topography.

We have added additional forcing terms that lead to a closed-form analytical solution. The equations are

$$\frac{\partial h}{\partial t} + \frac{\partial uh}{\partial x} = f_h \quad \text{and} \tag{4a}$$

$$\frac{\partial uh}{\partial t} + \frac{\partial uuh}{\partial x} = -gh\frac{\partial h}{\partial x} + uf_h + hf_u = F, \tag{4b}$$

where h is the height of the fluid, u is the velocity of the fluid, g is the acceleration due to gravity, and

$$f_h = (a_h\omega - a_u h_o k)\sin(kx + \omega t) + a_h a_u k \sin 2(kx + \omega t)$$

and

$$f_u = (a_h g k - a_u\omega)\sin(kx + \omega t) - 0.5a_u^2 k \sin 2(kx + \omega t);$$

Eqs. (4) have the analytic solutions

$$u = a_u \cos(kx + \omega t), \quad \text{and} \quad h = h_o - a_h \cos(kx + \omega t),$$

where h_o is the average height of the initial wave, ω is the frequency of the wave, and k is the wavenumber. Note, the analytical solutions are used as initial conditions (set $t = 0$) and periodic boundary conditions are employed in this simple model.

Considerable flexibility exists with respect to the solution of (4) in the JFNK approach. In our work we have chosen to use the JFNK method to determine the appropriate time-centered cell-face advective velocities and forces such that (4) can be discretized in conservative form. Additionally, only one equation, (4a), is currently being solved for by the JFNK method with the current JFNK formulation specifically designed to replace the explicit inner loop of the MOA approach with an implicit solver. For a description of the discretized shallow-water equations utilizing the MOA approach see appendix. Because their outer loops are of roughly the same form, this allows for ease of comparison between the two different solution techniques within the same code. Specifically, the discretized equations used in the outer loop of our shallow-water code with the JFNK method active are as follows:

$$h_i^{n+1} = h_i^n + \text{MPDATA}(h_i^n + 0.5f_{h_i}^n, \alpha_{i\pm1/2}^{n+1}) + 0.5f_{h_i}^{n+1} \tag{5a}$$

$$uh_i^{n+1} = uh_i^n + \text{MPDATA}(uh_i^n + 0.5F_i^n, \alpha_{i\pm1/2}^{n+1}) + 0.5F_i^{n+1}, \tag{5b}$$

where α is the advective cell-face velocity computed by the JFNK method and i is the spatial location. Here MPDATA refers to the second-order-accurate advection scheme of Smolarkiewicz (1984). Following Smolarkiewicz and Margolin (1993), additional terms of the form; $0.5f_{h_i}^n$ and $0.5F_i^n$, are added to h_i^n and uh_i^n, respectively, before the advective MPDATA algorithm is used to ensure second-order accuracy of the entire algorithm.

Assuming an inner loop of the form

$$F_{\text{sw}}(\hat{\mathbf{h}}^{n+1}) = \hat{h}_i^{n+1} - h_i^n + \text{DONOR}(h_i^n, \alpha_{i\pm1/2}^{n+1}) + f_{h_i}^{n+1} \tag{6}$$

with $\hat{\mathbf{h}}^{n+1}$ the state vector containing the shallow-water height field that satisfies the right-hand side of (6) where the hat symbol specifies a distinct inner loop variable, and DONOR referring to the standard (first-order accurate) donor-cell advection scheme, the appropriate forces and advective velocities for use in the second-order accurate outer loop (5) are obtained. Though linear stability theory suggests that breaking (4a) up into an advective component $u(\partial h/\partial x)$ and a divergent component $h(\partial u/\partial x)$ and implicitly solving for only the divergent component is stable, numerical testing of our nonlinear models revealed that for a model written in conservative form instabilities develop if a discretized form other than (6) is used. This was found to be especially true in simulations employing our Navier–Stokes model over regions of steep topography. Note, that the first-order pass of MPDATA is simply donor-cell advection with subsequent iterations of MPDATA presumably not effecting the stability (Smolarkiewicz 1984). Hence, (6) is not in a classic elliptic-like equation form and though the vector $F_{\text{sw}}(\hat{\mathbf{h}}^{n+1})$ represents only one equation, additional equations such as (5b) could easily be included in the vector. The ability to implicitly solve nontraditional elliptic equations is one of the virtues of the JFNK method.

In (6) $\alpha_{i\pm1/2}^{n+1}$ is formulated from the advective form of (5b). Specifically the left cell-face advective velocity is calculated as follows:

$$\alpha_{i-1/2}^{n+1} = 0.5(\hat{u}_i^{n+1} + \hat{u}_{i-1}^{n+1})\frac{\Delta t}{\Delta x}, \tag{7}$$

where,

$$\hat{u}_i^{n+1} = \hat{u}_i^n - 0.5g\frac{\Delta t}{\Delta x}(\hat{h}_{i+1}^{n+1} - \hat{h}_{i-1}^{n+1}), \tag{8}$$

with $\hat{u}_i^n = u_i^n + \text{UPWIND}(u_i^n, u_i^n(\Delta t/\Delta x)) + f_{u_i}^{n+1}\Delta t$ with UPWIND referring to the first-order upwind advective scheme or simply a nonconservative form of donor-cell advection. In the evaluation of (6) $\hat{\mathbf{h}}^{n+1}$ is computed using the methods put forth in section 2a so as to minimize the L_2 norm of (6) to a specified ϵ_{nk}. Note, when convergence is reached the force, F_i^{n+1}, is computed for use in (5b). Though (6) is linear, the above example will serve as a prototype for the nonlinear Navier–Stokes equations. In summary, the temporal advancement of (5) proceeds in the following steps:

1) $\alpha_{i\pm1/2}^{n+1}$, and F_i^{n+1} are computed within the inner loop using the JFNK approach;
2) $0.5F_i^n$ and $0.5f_{h_i}^n$ are added to uh_i and h_i, respectively;
3) uh_i and h_i are next updated using the second-order advective scheme MPDATA, and
4) $0.5F_i^{n+1}$ and $0.5f_{h_i}^{n+1}$ are then added to uh_i and h_i, respectively, to complete the time-stepping procedure.

b. *Preconditioning*

As mentioned in the previous sections, the preconditioning of a Krylov solver can lead to a significant reduction in the number of Krylov iterations and thus a reduction in overall computation time. In this section we will briefly demonstrate how a preconditioner is formed in the JFNK shallow-water framework.

For the one-dimensional shallow-water model and for preconditioners for which all elements of the Jacobian are calculated, the components of $\mathbf{P}$ are obtained numerically. To accomplish this, (6) was coded such that $F_{\text{sw}}(\mathbf{h}^n)$ could be computed on a point by point basis, $F_{\text{sw}}(h_i^n)$. The finite-difference stencil of (6) requires that five components for each entry of the preconditioning matrix be computed. The five entries on each row are

$$P_i^1 = \frac{\partial F_{sw}(\mathbf{h}^n)}{\partial h_{i-2}} = \frac{F_{sw}(\mathbf{h}^n + \epsilon_c \mathbf{e}_{i-2}) - F_{sw}(\mathbf{h}^n)}{\epsilon_c} \quad (9a)$$

$$P_i^2 = \frac{\partial F_{sw}(\mathbf{h}^n)}{\partial h_{i-1}} = \frac{F_{sw}(\mathbf{h}^n + \epsilon_c \mathbf{e}_{i-1}) - F_{sw}(\mathbf{h}^n)}{\epsilon_c} \quad (9b)$$

$$P_i^3 = \frac{\partial F_{sw}(\mathbf{h}^n)}{\partial h_{i}} = \frac{F_{sw}(\mathbf{h}^n + \epsilon_c \mathbf{e}_{i}) - F_{sw}(\mathbf{h}^n)}{\epsilon_c} \quad (9c)$$

$$P_i^4 = \frac{\partial F_{sw}(\mathbf{h}^n)}{\partial h_{i+1}} = \frac{F_{sw}(\mathbf{h}^n + \epsilon_c \mathbf{e}_{i+1}) - F_{sw}(\mathbf{h}^n)}{\epsilon_c} \quad (9d)$$

$$P_i^5 = \frac{\partial F_{sw}(\mathbf{h}^n)}{\partial h_{i+2}} = \frac{F_{sw}(\mathbf{h}^n + \epsilon_c \mathbf{e}_{i+2}) - F_{sw}(\mathbf{h}^n)}{\epsilon_c}, \quad (9e)$$

with $\epsilon_c = 1.0 \times 10^{-8}$ and $\mathbf{e}_i$ being the vector with all zero entries except at the location i where it has a value of one. Note that the coefficients for the preconditioner are currently computed only once at the start of the Newton iteration loop. Fortunately, for this particular problem, the Jacobian matrix has a simple form and can almost be exactly inverted using a five-point pentadiagonal solver algorithm with only the cyclic boundary conditions not being properly incorporated into the simple preconditioner. This preconditioner can be expressed at each point as follows:

$$P_i^1 q_{i-2} + P_i^2 q_{i-1} + P_i^3 q_i + P_i^4 q_{i+1} + P_i^5 q_{i+2} = r_i. \quad (10)$$

Boundary conditions are imposed by moving the appropriate terms to the right-hand side of (10). For example, at $i = 1$, $P_i^1 q_{i-2}$ and $P_i^2 q_{i-1}$ are both moved to the right-hand side. With this preconditioner active only one to two Krylov iterations per time step typically are required for convergence to be reached. Without this preconditioner active, 20 or more Krylov iterations are sometimes required for convergence.

Also, from our earlier discussion in section 2b one can see that equation (10) can be approximately inverted in a matrix-free fashion by only storing the diagonal (P_i^3) portion of the matrix. Results from this type of preconditioner as well as the previously mentioned preconditioner will be presented in the next section.

c. Results

The JFNK approach is most appropriate for regimes where the flow velocity is much less than the speed of the fastest wave. But an important question that will be at least partially addressed in this section is, how much less? Before quantifying this regime, we need to first demonstrate that for regimes of interest the JFNK algorithm is of comparable accuracy to the MOA algorithm. For the first series of tests we have chosen the following parameters for our simple model. $g = 1.0$ m s^{-2}, $h_o = 1.0$ m, $a_h = 0.01$ m, $a_u = 0.01$ m s^{-1}, $k = 2$ m^{-1}, and $\omega = 0.0$ s^{-1} (a steady-state forcing regime), which gives $V_g = \sqrt{g(h_o + a_h)} = \sqrt{1.01}$ m s^{-1} where V_g is the speed of the fastest gravity wave in this model. For the MOA simulation, MOAS1, $N = 50$ subcycles were used in the inner loop such that $N(\Delta t/\Delta x) = 50.0$ s m^{-1}; whereas, for the JFNK simulation, JFNKS1, $\Delta t/\Delta x = 50.0$ s m^{-1} was specified. Note, for the MOA algorithm $V_g \Delta t/\Delta x$ of the inner loop must be no larger than about one (see Fig. 1 of Reisner et al. 2000) for stability of the inner loop. Following Smolarkiewicz (1984), we ran the simulations at successively finer resolution while maintaining a fixed Courant number $a_u(\Delta t/\Delta x)$ with each successive Δt and Δx being reduced by a factor of 2. For each simulation, we calculate the truncation error,

$$\text{trunc\_error}(N_T, N_X) = \sqrt{\frac{\sum_{i=1}^{N_X} (\psi(T, x_i) - \psi_i^{N_T})^2}{N_T N_X}}, \quad (11)$$

where $N_T = TN/\Delta t$ with N equal to the number of subcycles within the inner loop ($N = 1$ for JFNK simulations), $N_X = L/\Delta x$ with $T = 2\pi$ s and $L = 2\pi$ m, and $\psi(T, x_i)/\psi_i^{N_T}$ are the analytical and numerical solutions, respectively, at the point (T, x_i). Initially $N_X = N_T = 18$, with this number repeatedly being doubled. Also, in all JFNK simulations a value of 1×10^{-2} is used for γ from (3).

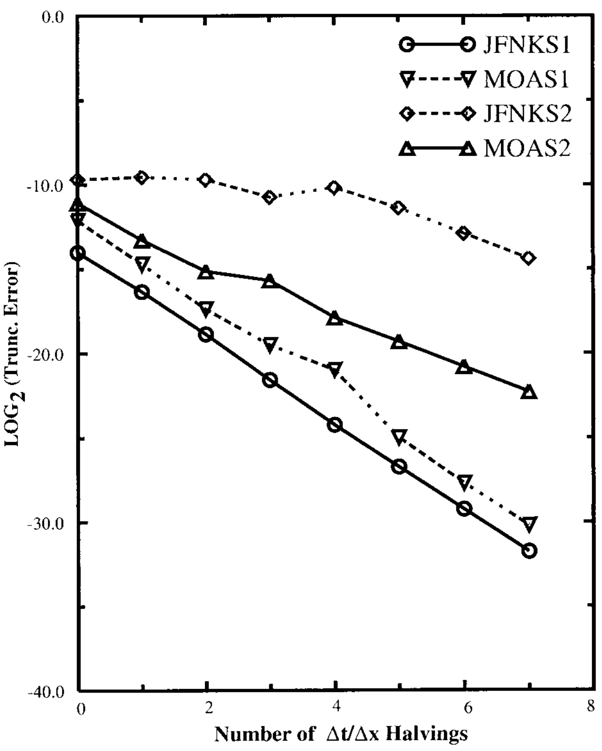

FIG. 1. The dependence of the measure of the truncation error on the number of halvings of the grid and time increments with the lines labeled JFNKS1 (S2) and MOAS1 (S2) from simulations with ω = 0.0 (1.0), where ω is the frequency of the applied forcing.

Figure 1 shows that for the current choice of parameters, the JFNK approach produces a solution that is slightly more accurate than the MOA technique; however, the slopes of the lines still indicate second-order convergence in time [cf. with convergence curves shown in Smolarkiewicz (1984)]. In JFNKS1 $\epsilon_{nk} = 1.0 \times 10^{-8}$, a reduction to $\epsilon_{nk} = 1.0 \times 10^{-3}$ produced an error curve (not shown in Fig. 1) that lies between the two curves of JFNKS1 and MOAS1.

Numerical tests suggest that for this particular regime an $\epsilon_{nk} > 1 \times 10^{-3}$ prohibits second-order convergence in time and space. Surprisingly, changing ω from 0.0 to 1.0 s^{-1} with all other parameters fixed, results in a reversal in accuracy between the two numerical approaches. Not only is the MOA solution, MOAS2, more accurate than the JFNK solution, JFNKS2, the slope of the line from MOAS2 is somewhat less than the slope of the line from JFNKS2. The ability of the MOA approach to resolve the movement of the fastest waves—though not necessarily the amplitude—in the inner loop is the principal reason for why the MOA algorithm is more accurate in this flow regime. Also, in agreement with the above reasoning, additional simulations reveal that when the flow velocity nears the speed of the fastest wave, the MOA technique becomes more accurate than the JFNK approach. However our tests do suggest that the JFNK approach maintains second-order accuracy in regimes for which it was designed, $a_u/V_g \ll 1$, and low-frequency forcing.

Though we have demonstrated the use of the JFNK method in a conservative shallow-water equation set employing the forward-in-time advection scheme MPDATA, we could have just as easily used a more traditional atmospheric algorithm, for example, the second-order accurate leapfrog advection scheme employed in a nonconservative shallow-water framework. The function to be minimized in this framework is simply

$$F^{sw}(\mathbf{h}^{n+1}) = h_i^{n+1} - h_i^{n-1} + \mathrm{LEAP}\left(h_i^n, u_i^n\frac{\Delta t}{\Delta x}\right) + h_i^{n+1}\frac{\Delta t}{\Delta x}(u_{i+1}^{n+1} - u_{i-1}^{n+1}) - 2.0 f_h \Delta t$$

with LEAP referring to the standard leapfrog advection scheme and the divergent velocity field being represented by $u_i^{n+1} = u_i^{n-1} - \mathrm{LEAP}[u_i^n, u_i^n(\Delta t/\Delta x)] - g(\Delta t/\Delta x)(h_{i+1}^{n+1} - h_{i-1}^{n+1}) + 2.0 f_u \Delta t$. Note that the divergence term can easily be linearized by replacing h^{n+1} with h^n and enforcing only one Newton iteration per time step. The effect of this linearization was found to lead to a less accurate solution (not shown) then from the nonlinear leapfrog model for $\omega > 0.0$.

Next, for a fixed forcing we attempt to quantify the point at which the ratio (MOA–JFNK) of accuracy time efficiency for the two numerical approaches tips in favor of the JFNK method. For these simulations a_u/V_g was varied from 0.0001 to 0.5 with all other parameters the same as used in JFNKS1, except that all simulations were run for a fixed 2000 time steps and either the number of subcycles for the MOA approach or $\Delta t/\Delta x$ for the JFNK approach was chosen such that $(\Delta t/\Delta x)a_u = 0.5$. The ratio (MOA–JFNK) of accuracy times efficiency for the the two numerical approaches is illustrated by the solid line in Fig. 2 and clearly shows that the cross-over point for which implicit techniques become viable is for flow regimes around $a_u/V_g = 0.02$. Obviously preconditioning, which influences the efficiency of the JFNK approach, should shift this crossover point towards the right. Indeed with the pentadiagonal solver based preconditioner active the dashed line in Fig. 2 indicates a rightward movement of the crossover point to around $a_u/V_g = 0.04$. Note, that this crossover point is only approximate, factors such as the weighting functions used in the MOA procedure or the type of preconditioner used in the JFNK method can have a significant impact on the location of the crossover point.

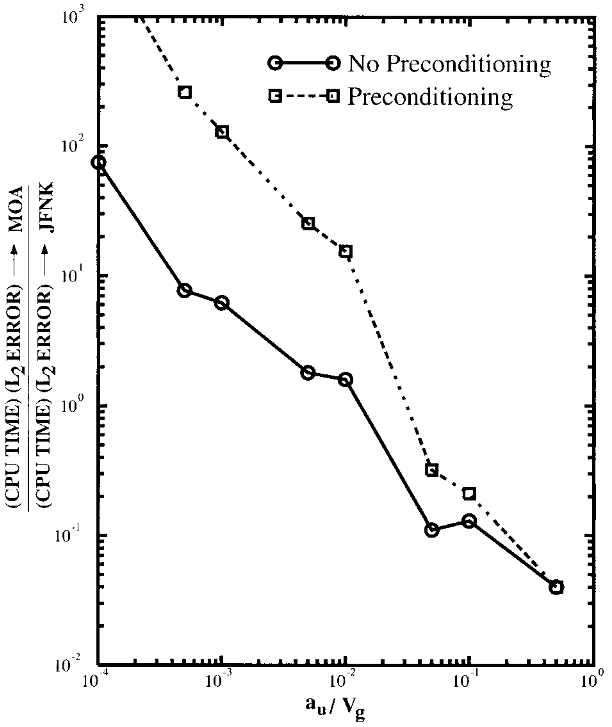

FIG. 2. Ratio of CPU time multiplied by the L_2 norm error for MOA simulations over JFNK simulations as a function of a_u/V_g with a_u/V_g being the relative speed of the material motion with respect to the speed of the fastest wave. Ratios of CPU time multiplied by the L_2 norm error less than one imply a regime where the explicit MOA algorithm is both more efficient and accurate than the implicit JFNK alogorithm. JFNK simulations with the preconditioner active employ the nearly direct ADI preconditioner.

Finally, we will consider the performance of a few simple matrix-free preconditioners in the shallow-water environment. Matrix-free preconditioners in this framework can be rather easily formulated based upon the methodology put forth at the end of section 2. The model setup is the same as used to generate the curves in Fig. 1 for $\omega = 0$. Results are presented for two matrix-free (MF) preconditioners: MF Jacobi and MF symmetric Gauss–Seidel (SGS). In Table 1 the results for these two matrix-free preconditioners are compared to the matrix versions of Jacobi and SGS as well as no preconditioning and an ''almost'' direct solver for comparison. Note that the direct solver is our pentadiagonal solver, which does not capture all of the coupling due to the periodic boundary conditions. From Table 1 it is clear that the MF preconditioners provide almost the same level of Krylov iteration reduction but at a slightly higher CPU cost. Therefore, the MF preconditioners allow the option of trading CPU time for storage cost. This finding is consistent with that of Qin et al. (2000).

Thus, in this section we have attempted to provide some guidance as to what type of preconditioner to use as well as when implicit solvers become viable with respect to explicit flow solvers in terms of both accuracy and efficiency. Our results suggest that in flow regimes in which the normalized velocity component is about 50 times less than the speed of the fastest wave, implicit solvers appear to be the ideal choice. It should be noted that most mesoscale weather prediction codes, such as the fifth-generation Mesoscale Model (Grell et al. 1994) or the Regional Atmospheric Modeling System (Pielke et al. 1992), take advantage of the natural disparity of scales between horizontal and vertical motions and employ explicit solvers in the horizontal and implicit solvers in the vertical. Unfortunately, this approach does not lead to an entirely universal solver, one that is accurate for all scales, and obviously can, under certain circumstances, lead to large splitting and/or conservation errors. Though not as potentially efficient as the approach used in mesoscale weather prediction models, we have chosen to implement the JFNK approach into a model, HIGRAD (short for high gradient flow solver), which allows the fully compressible Navier–Stokes equations to be solved implicitly in a three-dimensional framework. Of course, with the inclusion of buoyancy the current nonlinear shallow-water model could rather easily mimic the vertical implicit solver used in most mesoscale models.

4. HIGRAD

a. Navier–Stokes model

HIGRAD was initially developed at the Los Alamos National Laboratory (Reisner et al. 2000) to study the movement of wildfires. In this flow regime, the MOA approach is a good choice; however, to extend HIGRAD to flow situation in which the MOA technique is no longer computational efficient the JFNK method has been implemented into HIGRAD. Details of the implementation of the JFNK approach into HIGRAD will be discussed in this section.

HIGRAD solves the compressible form of the Navier–Stokes equations. The equations written in flux form are

$$\frac{\partial G u\rho}{\partial t} + \nabla \cdot (\mathbf{v} G\rho u) = G R_{u\rho} \qquad (12a)$$

$$\frac{\partial G v\rho}{\partial t} + \nabla \cdot (\mathbf{v} G\rho v) = G R_{v\rho} \qquad (12b)$$

$$\frac{\partial G w\rho}{\partial t} + \nabla \cdot (\mathbf{v} G\rho w) = G R_{w\rho} \qquad (12c)$$

$$\frac{\partial G \theta\rho}{\partial t} + \nabla \cdot (\mathbf{v} G\rho \theta) = 0 \qquad (12d)$$

$$\frac{\partial G \rho}{\partial t} + \nabla \cdot (\mathbf{v} G\rho) = 0 \qquad (12e)$$

$$p = \frac{(R_d \rho\theta)^{C_p/C_v}}{p_o^{R_d C_v}}, \qquad (12f)$$

where u, v, and w are the velocity components in the coordinate system $(x, y, z) = (x_c, y_c, z_c)$ with the subscript c referring to Cartesian coordinates, θ is the potential temperature, $\theta = T(p_o/p)^{R_d/C_p}$ with T the temperature of the gas and $p_o = 10^5$ N m^{-2} the base state pressure, and

$$G = \det(\partial \mathbf{x}_c/\partial \mathbf{x}) = [\det(G^{IJ})]^{-1/2}$$

is the Jacobian of transformation with

$$G^{I,J} = \sum_{K=1}^{3} \frac{\partial x^I}{\partial x_c^K}\frac{\partial x^J}{\partial x_c^K}.$$

The equation of state, (12f), relates the total pressure, p, to the density, ρ, and the potential temperature, θ, of the gas.

The constants, $C_p/C_v = 1004/717$ J K^{-1} kg^{-1}, and $R_d = 287$ J K^{-1} kg^{-1}, in (12f) are the specific heat of air at constant pressure/volume and the gas constant of dry air, respectively. The contravariant vertical component of the advective velocity vector $\mathbf{v} = u\hat{\mathbf{i}} + v\hat{\mathbf{j}} + \omega\hat{\mathbf{k}}$ appears as the result of employing a terrain-following coordinate system, $(x, y, z) = [x_c, y_c, H(z_c - h)/(H - h)]$, where H is the model depth and $h = h(x_c, y_c)$ the model bottom. It is related to the Cartesian velocity components by the relationship, $\omega = G^{13}u + G^{23}v + G^{-1}w$.

The forces $R_{u\rho}$, $R_{v\rho}$, and $R_{w\rho}$ in (12) are

$$R_{u\rho} = -\frac{\partial p'}{\partial x} - G^{13}\frac{\partial p'}{\partial z} + f\rho(v - v_e) - \hat{f}\rho(w - w_e) \qquad (13a)$$

$$R_{v\rho} = -\frac{\partial p'}{\partial y} - G^{23}\frac{\partial p'}{\partial z} - f\rho(u - u_e) \qquad (13b)$$

$$R_{w\rho} = -G^{-1}\frac{\partial p'}{\partial z} - \rho' g + \hat{f}\rho(u - u_e), \qquad (13c)$$

where u_e, v_e, and w_e are the balanced environmental velocity components; $f = 2\Omega \sin\varphi$ and $\hat{f} = 2\Omega \cos\varphi$ are the z and y components of the earth's rotation axis at the latitude φ; g is the acceleration due to gravity; $\rho' = \rho - \rho_e$ is the density perturbation where $\rho_e = \rho_e(z_c)$ is the environmental density; and $p' = p - p_e$ is the pressure perturbation with $p_e = p_e(z_c)$ the environmental pressure.

The basic algorithm for integrating (12) on a discrete mesh is second-order accurate in space and time. We use a grid in which all variables are defined at the same location. Employing the JFNK approach, the discretized equations of the outer loop of HIGRAD are

$$u\rho_i^{n+1} = \mathrm{MPDATA}(u\rho_i^n + 0.5R_{u\rho_i}^n,\ \alpha_{i\pm1/2}^{n+1},\ G_i) + 0.5R_{u\rho_i}^{n+1} \tag{14a}$$

$$\upsilon\rho_i^{n+1} = \mathrm{MPDATA}(\upsilon\rho_i^n + 0.5R_{\upsilon\rho_i}^n,\ \alpha_{i\pm1/2}^{n+1},\ G_i) + 0.5R_{\upsilon\rho_i}^{n+1} \tag{14b}$$

$$w\rho_i^{n+1} = \mathrm{MPDATA}(w\rho_i^n + 0.5R_{w\rho_i}^n,\ \alpha_{i\pm1/2}^{n+1},\ G_i) + 0.5R_{w\rho_i}^{n+1} \tag{14c}$$

$$\theta\rho_i^{n+1} = \mathrm{MPDATA}(\theta\rho_i^n,\ \alpha_{i\pm1/2}^{n+1},\ G_i) \tag{14d}$$

$$\rho_i^{n+1} = \mathrm{MPDATA}(\rho_i^n,\ \alpha_{i\pm1/2}^{n+1},\ G_i), \tag{14e}$$

where i is shorthand for the spatial location in three dimensions. In the Navier–Stokes model the second-order advection scheme MPDATA employs the synchronous transport scheme of Schär and Smolarkiewicz (1996) to ensure that scalar variables preserve monotonicity.

As was the case in the shallow-water equation set, to compute α and the forces from the JFNK method the continuity equation is solved to a specified nonlinear residual. The discretized form chosen for this equation is the following:

$$F_\rho(\hat{\boldsymbol{\rho}}^{n+1}) = \hat{\rho}_i^{n+1} - \rho_i^n + \mathrm{DONOR}(\rho_i^n,\ \alpha_{i\pm1/2}^{n+1}), \tag{15}$$

with $\hat{\boldsymbol{\rho}}^{n+1}$ being a state vector that satisfies the right-hand side of (15). Forming the velocities required to compute the cell-face velocities, $\alpha_{i\pm1/2}^{n+1}$, requires a straightforward algebraic inversion of the advective form of (12) with respect to ρ, u, υ, and w—and the formulation of the boundary value problem for p'. Following Smolarkiewicz and Margolin (1997) the inversion of (12) produces

$$\mathbf{V}^I = \varepsilon\left(\nu^I - \sum_{J=1}^{3} C^{IJ}\frac{\partial t}{\rho}\frac{\partial p'}{\partial x^J}\right), \tag{16}$$

where $\mathbf{V}^I \equiv (u, \upsilon, \omega)$ and the coefficients ϵ, ν (note our buoyancy is of different form), and C^{IJ} are provided in appendix A of Smolarkiewicz and Margolin (1997). Next, the individual components of $\mathbf{V}^I$ are discretized, multiplied by G_i, averaged in space, and then multiplied by $\Delta t/\Delta x^I$ to obtain the appropriate cell-face advective velocity. The pressure perturbation needed to compute the pressure gradient forces is obtained using a potential temperature field that is calculated prior to the start of the Newton iteration loop from the advective form of (12d). As in the simple shallow-water model, advection of both potential temperature and of all the velocity fields is computed with a first-order upwind scheme. Hence, like in the shallow-water model, the JFNK method is used to compute advective velocities and forces in a first-order manner for use in the second-order outer loop. In this study, we have chosen to include only one equation in the nonlinear functional; however, we have experimented with including additional equations such as (12d) or all of (12) into the algorithm. Second-order advective schemes could as well be included in the functional eliminating the need for the outer loop and allowing for the implicit calculation of advection.

As was noted earlier, preconditioning can lead to a significant reduction in the number of Krylov iterations taken per Newton iteration. Unfortunately, unlike the shallow-water framework, a preconditioner that directly inverts the Jacobian matrix cannot realistically be developed for the current sparse matrix. Instead we have experimented with approximate and simple preconditioners such as Jacobi, SSOR, ILUT (an incomplete LU with threshold; Saad 1995), and the five-point pentadiagonal solver used in the shallow-water model, but extended to multidimensions, producing an ADI solver. Currently, all terms within the preconditioning matrix are computed using a scalar form of (15). In two (three) dimensions 13 (21) terms are required to form the preconditioner. If computer memory is a concern, Jacobian terms involving terrain transformations can be dropped [four (eight) terms in two (three) dimensions] from the preconditioner and/or matrix-free approximations can be used; however, as will be shown for steep mountains the number of Krylov iterations does increase significantly if any terms are indeed dropped. This finding is in disagreement against what was found by Skamarock et al. (1997) regarding the inclusion of terrain terms in their preconditioner, but in their ‘‘nonhydrostatic example’’ the slope of the mountain was still relatively small. Additionally, for most atmospheric flow regimes the coefficients of the preconditioner do not change significantly and hence need not be computed every time step. In fact, for the mountain flow problems to be shown next, little if any, decrease in the total number of Krylov iterations was found when the coefficients were computed only once, at the beginning of the simulations. It is important to emphasize that the approximations made in the preconditioner do not effect the accuracy of the JFNK method since these approximations only effect the convergence rate of the linear problem, not the residual evaluations.

b. Results

As an illustration of the wide range of problems for which our Navier–Stokes solver is relatively efficient we will demonstrate its use on two simple problems, a hydrostatic mountain wave problem and a nonhydrostatic mountain wave problem with a very steep mountain. For both problems numerous simulations were run to determine, for example, how sensitive convergence rates were to the type of a preconditioner used or to the number of iterations used in a preconditioner. The setup used for our hydrostatic mountain wave problem is the same as that employed by Skamarock et al. (1997). This test problem setup assumes uniform flow $U_o = 20$ m s^{-1}, constant Brunt–Väisälä frequency $N_v = 0.02$ s^{-1}, and a bell-shaped mountain centered in the middle of the domain,

$$h(x) = H[1 + (x/L)]^{3/2}, \qquad (17)$$

with a horizontal scale $L = 50 \times 10^3$ m and height $H = 1000$ m. The computational domain is covered by 100 grid points in the horizontal and 45 grid points in the vertical. The horizontal resolution is 10^3 m and the approximate vertical resolution is 420 m. Simulations were run with a time step of 50 s for 800 time steps. Lateral and top gravity wave absorbers were active during a simulation. For this test problem multidimensional forms of the ADI preconditioner or an SSOR preconditioner were used, but in agreement with the findings of Skamarock et al. (1997) we found little difference in the total number of Krylov iterations if ‘‘only’’ a vertical form of either preconditioner with terrain terms excluded was used. Figure 3 reveals the average number of Krylov iterations per time step as a function of the specified nonlinear residual utilizing either the ADI preconditioner employing one iteration or the SSOR preconditioner employing eight iterations. For the ADI preconditioner two sets of curves are shown with the results being generated by simulations employing different γ, the parameter used in (3). As evident in Fig. 3, for this flow regime the ADI preconditioner is more robust than the SSOR preconditioner, especially for smaller ϵ_{nk} and hence appears to be the better of the two preconditioners. Significant differences in the total number of Krylov iterations were also found when the same preconditioner is used but with a different γ. For the larger γ the number of Krylov iterations was reduced by about one-half, but the number of Newton iterations (not shown) per time step increased by about a factor of 2. However, only small differences in computational timings were found between the two sets of simulations employing the different γ's. In comparison with the total number of Krylov iterations from the linear solver used in Skamarock et al., the total number of Krylov iterations from the JFNK solver with $\gamma = 0.5$ (0.1) were fewer (greater) than produced by their solver.

The results from the previous section on the shallow-water model suggest that for this flow regime the JFNK solver should be more efficient than the MOA solver. Indeed, for $\epsilon_{nk} = 10^{-6}$ and $\gamma = 0.1$ the JFNK solver was about three times faster than the MOA solver. In addition to significant differences in timing, Fig. 4 reveals that differences in flow features exist at a given time with respect to the structure of the nonlinear mountain wave. These differences are to be expected in the lee (Doyle et al. 2000) and appear to be the result of the random breaking of mountain waves in the lee. Additionally, we have conducted simulations comparing the nonlinear model against a more traditional linear model. The linear model was achieved by allowing only one Newton iteration per time step, by setting the time level of the density in the pressure gradient term to be at n, and by breaking up the discretized form for pressure as $p = (R_d\rho_i^{n+1}\theta_i^{n+1})(R_d\rho\theta_i^n)^{[(C_p/C_v)-1.]}/p_o^{R_d/C_v}$. Computing L_2 norm error measures of the u velocity field at 40 000 s from both the linear and nonlinear solvers and using as an ‘‘exact’’ solution results from a simulation employing a very small time step, 0.025 s, revealed the nonlinear solver to be about 10% (1%) more accurate than the linear solver for a time step of 50 (6.25) s. Based upon this result we hypothesize that differences between a nonlinear solver versus linear solver would be even larger if additional physics, such as condensation, were included in the equation set.

Though the results from the previous test problem suggest that HIGRAD could be used efficiently in that flow regime, HIGRAD was primarily designed to ex-

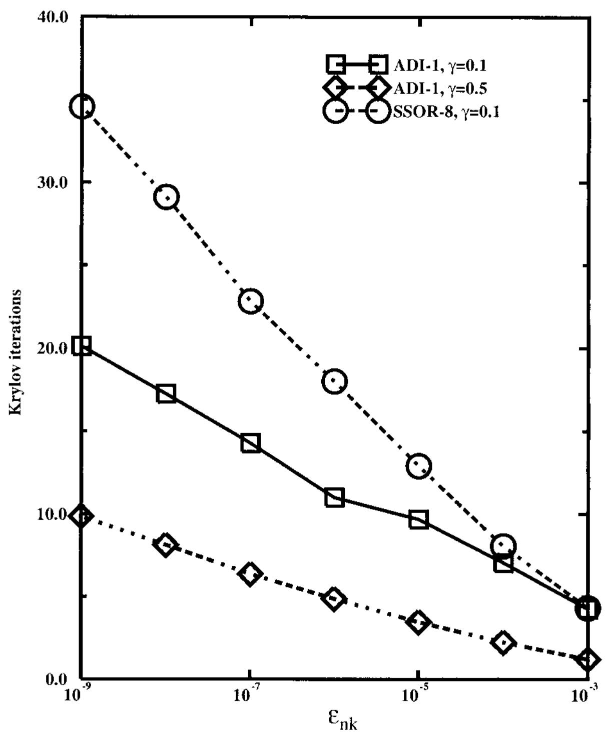

FIG. 3. Average number of Krylov iterations as a function of ϵ_{nk} for the hydrostatic-scale mountain wave simulations employing the following preconditioners: (a) ADI preconditioner with one iteration and $\gamma = 0.1$ denoted by solid line with square data points, (b) ADI preconditioner with one iteration and $\gamma = 0.5$ denoted by dashed line and diamond data points, and (c) SSOR preconditioner with eight iterations denoted by dashed line and round data points.

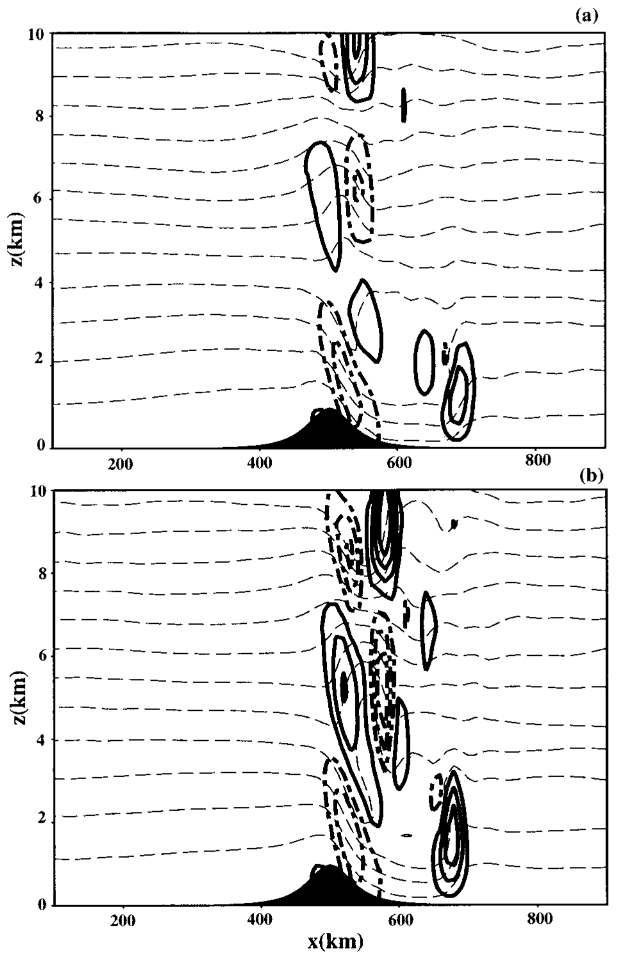

FIG. 4. Hydrostatic-scale mountain wave simulation at 40 000 s with panels (a) and (b) from the JFNK and MOA methods, respectively. Potential temperature, θ, is contoured in light dashed lines with an interval of 12 K. Contours of vertical velocity are in black with an interval of 0.4 m s^{-1} with solid (dashed) lines indicating areas of rising (sinking) motions. The plotted region is 1000 km × 10 km with the specified Froude number being 1.0.

TABLE 1. Statistics from various preconditioners.

Preconditioner	Newton iterations per time step	Krylov iterations per time step	Total CPU (s)
No preconditioning	3.99	8.838	24.5
One-pass Jacobi	3.98	8.510	28.2
One-pass MF Jacobi	3.98	8.510	35.8
Three-pass Jacobi	3.98	5.006	19.9
Three-pass MF Jacobi	3.98	5.006	33.5
One-pass SGS	2.99	4.327	12.5
One-pass MF SGS	2.99	4.329	17.8
Three-pass SGS	2.49	3.203	11.2
Three-pass MF SGS	2.49	3.202	20.2
Direct solve	1.00	1.994	4.4

amine flow features at spatial resolutions of tens of meters or less. As such, we are primarily concerned whether our preconditioners will be robust enough to make the JFNK method efficient in this type of flow regime. To address this issue we have set up a test problem that utilizes 20-m spatial resolution and a steep bell-shaped mountain with a height of 300 m and a horizontal scale of 100 m. This test problem setup assumes uniform flow $U_o = 2$ m s^{-1}, constant Brunt–Väisälä frequency $N_v = 0.02$ s^{-1}, and the same number of nodal points as used in the previous test problem. Simulations were run with a time step of 1 s for 900 time steps. Once again either a multidimensional ADI preconditioner or an SSOR preconditioner was used in the simulations with γ being set to 0.1. Of note, for this flow regime, both preconditioners tend to become less efficient, especially the ADI preconditioner, as the number of processors increases. Hence, when the code is run in parallel more sophisticated preconditioners, such as a two-level multigrid method may be required. This research topic will be deferred for later papers. Figure 5 shows that for this flow regime the SSOR preconditioner employing eight iterations is, unlike the previous problem, more robust in reducing the number of Krylov iterations than the ADI preconditioner with two iterations. Increasing the

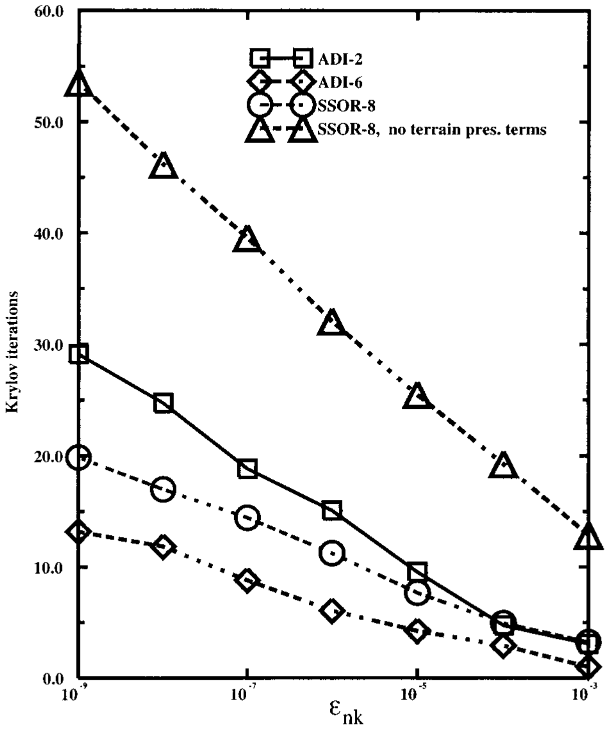

FIG. 5. Average number of Krylov iterations as a function of ϵ_{nk} for the nonhydrostatic-scale mountain wave simulations employing the following preconditioners: (a) ADI preconditioner with two iterations denoted by solid line with square data points, (b) ADI preconditioner with six iterations denoted by dashed line and diamond data points, (c) SSOR preconditioner with eight iterations denoted by dashed line and round data points, and (d) SSOR preconditioner with eight iterations and terrain terms "turned off" denoted by dashed line and triangle data points.

number of ADI iterations from two to six does reduce the number of Krylov iterations below that of the SSOR preconditioner; however, the computational cost of simulations employing the ADI preconditioner with six iterations is greater than simulations employing the SSOR preconditioner with eight iterations. Similar findings with respect to the utility of using successive over relaxation (SOR) as a preconditioner in both a serial and parallel environment were found by Delong and Ortega (1995 and 1998). Note that, we have run simulations employing the SSOR preconditioner without the terrain terms active in the preconditioner and found the number of Krylov increases by a factor of 2 (top line in Fig. 5). This finding suggests that for steep terrain the extra terms related to terrain transformations should be included. Also, even with the terrain terms active in the preconditioner a simulation with $\epsilon_{nk} = 10^{-6}$ and the SSOR preconditioner active is still slightly more expensive than a simulation employing the MOA approach. This is not unexpected since the current regime ($U/V \approx 0.015$) is near the cutoff point shown in Fig. 2.

5. Concluding remarks

The mechanics of the JFNK method have been illustrated in this paper. The method has been applied to the conservative Navier–Stokes equations and shown to be an efficient solver, comparable in efficiency to linearized solvers. But, unlike linearized solvers, the JFNK method enables nonlinear terms to be modeled without any compromise. In certain flow situations we believe that compromising accuracy by the linearization of certain terms could lead to a failure of a numerical model to correctly forecast intensification of a particular weather event. In this work we have chosen to concentrate on the details of the method and not the application. As such, rather simple problems were examined. Future papers will use this solver on more difficult flows such as occur in the vicinity of a wildfire or a hurricane. In this regime, strong nonlinear coupling occurs between terms representing combustion, latent heat release, diffusion, and advection. To capture all of these physical processes implicitly will require that more than one equation be solved for within the JFNK framework. Here preconditioning becomes even more of an issue and physics-based preconditioning will be employed that addresses the coupled physics in individual pieces in a divide and conquer strategy (Mousseau et al. 2000). Another approach would be to use the current solver as a preconditioner for a solver in which all terms in the Navier–Stokes equation set are handled implicitly.

The development of a fully implicit solver will not only benefit the simulation of hurricanes or wildfires, but also the simulation of other difficult problems in atmospheric science. For example, this solver could be used in the assimilation of atmospheric weather data, as well as being used for the prediction of a future atmospheric event. The ability to use the same solver for data assimilation and for prediction is an appealing aspect of the JFNK approach. Another potential use for the solver is in the simulation of global-scale flows. The small grid increments found near the poles considerably reduce the time step that can be used by an explicit advection scheme. By using the JFNK method to solve an Eulerian advection scheme implicitly, a potential savings in computational time might be the result; however, we believe that the current use of the JFNK solver to solve implicitly for advective velocities and forces in a first-order manner is a good compromise between efficiency and accuracy for most atmospheric flow situations.

A comparison between the explicit MOA solver and the implicit JFNK solver has revealed regimes appropriate for each solver. The JFNK solver tends to do best in regimes characterized by weak forcing and where material motions are much smaller than the speed of the fastest wave. In a predictive mode, the model should have the ability to switch between either solver depending on how the flow conditions evolve. How the frequency of switching influences stability of the model will need to be investigated in the future. Another example in which the solvers could be used in combination is to use the MOA algorithm to provide an initial guess for the JFNK solver. To help speed up the MOA algorithm so that an initial guess could be obtained in a timely manner, the speed of the fastest wave within the MOA method could be artificially reduced allowing for a larger time step to be used. We have conducted preliminary experiments with this procedure and they indicate some promise. Finally, use of a fully three-dimensional implicit solver in an operational weather forecasting environment is not without risks; occasionally, a Krylov solver may get stuck or require a large number of iterations to converge. For these situations, logic embedded within a code could allow a switch from the JFNK algorithm to the MOA algorithm. Likewise, whether or not a Krylov solver gets stuck is crucially dependent on the robustness of the preconditioner used in the Krylov solver. We have shown in this paper that under certain flow regimes either an ADI or SSOR preconditioner works best in terms of reducing the number of Krylov iterations. We believe that some combination of both would be preferable and we are currently testing a preconditioner that incorporates both approaches in a multigrid framework.

Acknowledgments. Los Alamos National Laboratory is operated by the University of California for the U.S. Department of Energy. The authors acknowledge helpful discussions with Piotr Smolarkiewicz, Len Margolin, Rodman Linn, and Bill Rider. We would also like to thank two anonymous reviewers who put considerable time in reviewing this manuscript, which ultimately led to substantial improvements in the revised text.

APPENDIX

MOA Algorithm: Further Details

Following Reisner et al. (2000) to solve (4) to second-order accuracy in time and space within the MOA framework, we discretized the outer loop of the MOA algorithm as follows:

$$h_i^{n+1} = h_i^n + \mathrm{MPDATA}(h_i^n, \overline{u}_{i\pm1/2}N\Delta t/\Delta x) + \mathrm{DONOR}(\overline{f}_{h_i}, \overline{u}_{i\pm1/2}N\Delta t/2\Delta x) \qquad \text{(A1a)}$$

$$uh_i^{n+1} = uh_i^n + \mathrm{MPDATA}(uh_i^n, \overline{u}_{i\pm1/2}N\Delta t/\Delta x) + \mathrm{DONOR}(\overline{F}_i, \overline{u}_{i\pm1/2}N\Delta t/2\Delta x), \qquad \text{(A1b)}$$

where $\overline{u}_{i\pm1/2} = 0.5(\overline{u}_i + \overline{u}_{i\pm1})$. The averaged quantities $\overline{u}$, $\overline{f}_h$, and $\overline{F}$ are computed in the inner loop by averaging in time using the expression

$$\overline{x}_i = \frac{\sum_{n=0}^{N} w_n \hat{x}^n}{N}, \qquad \text{(A2)}$$

with N the number subcycles of the inner loop, w_n the weighting coefficients that satisfy $w_0 + \cdot + w_N = N$, where $w_o = 0.3$, $w_1 = \cdot = w_{N-1} = 1$, $w_N = 0.7$, and $\hat{x}$ representing an inner loop variable that is to be averaged in time. Note, the advection of the averaged forcing terms by the donor-cell advection scheme is required to ensure second-order accuracy of the overall algorithm; see Smolarkiewicz and Margolin (1993).

The inner loop is discretized as follows:

$$\hat{h}_i^{n+1} = \hat{h}_i^n + \mathrm{DONOR}(\hat{h}_i^n, u_{A_{i\pm1/2}}^{n+1/2}) + \hat{f}_{h_i}^{n+1}\Delta t \qquad \text{(A3a)}$$

$$u\hat{h}_i^{n+1} = u\hat{h}_i^n + \mathrm{DONOR}(u\hat{h}_i^n, u_{A_{i\pm1/2}}^{n+1/2}) + \hat{F}_i^{n+1}\Delta t, \qquad \text{(A3b)}$$

where $u_{A_{i\pm1/2}^{n+1/2}} = 0.5(\hat{u}_i^{n+1/2} + \hat{u}_{i\pm1}^{n+1/2})\Delta t/\Delta x$; $\hat{u}^{n+1/2}$ is computed by a simple extrapolation in time.

To advance this system in time, one first solves (A3) for N time increments and computes the sums using (A2) to form the averaged quantities needed to solve (A1).

REFERENCES

Brown, P. N., and Y. Saad, 1990: Hybrid Krylov methods for nonlinear systems of equations. *SIAM J. Sci. Stat. Comput.,* **11,** 450–481.

Chan, T. F., and K. R. Jackson, 1984: Nonlinearly preconditioned Krylov subspace methods for discrete Newton algorithms. *SIAM J. Sci. Stat. Comput.,* **5,** 533–542.

Clark, T. L., 1977: A small-scale dynamic model using a terrain following coordinate transformation. *J. Comput. Phys.,* **24,** 186–215.

Delong, M. A., and J. M. Ortega, 1995: SOR as a preconditioner. *Appl. Numer. Math.,* **18,** 431–440.

——, and ——, 1998: SOR as a preconditioner II. *Appl. Numer. Math.,* **26,** 465–481.

Dembo, R. S., and T. Steilhaug, 1983: Truncated-Newton algorithms for large scale unconstrained optimization. *Math. Prog.,* **26,** 190–212.

——, S. C. Eisenstat, and T. Steilhaug, 1982: Inexact Newton methods. *SIAM J. Numer. Anal.,* **19,** 400–408.

Doyle, J. D., and Coauthors, 2000: An intercomparison of model-predicted wave breaking for the 11 January 1972 Boulder windstorm. *Mon. Wea. Rev.,* **128,** 901–914.

Eisenstat, S. C., and H. F. Walker, 1996: Choosing the forcing terms in a inexact Newton method. *SIAM J. Sci. Comput.,* **17,** 16–32.

——, H. C. Elman, and M. H. Schultz, 1983: Variational iterative methods for nonsymmetric systems of linear equations. *SIAM J. Numer. Anal.,* **20,** 345–357.

Grell, G. A., J. Dudhia, and D. R. Stauffer, 1994: A description of the fifth-generation Penn State/NCAR mesoscale model (MM5). NCAR/TN-398 + STR, 121 pp. [Available from National Center for Atmospheric Research, Boulder, CO 80307; and online at http://www.mmm.ucar.edu/mm5.]

Knoll, D. A., 1998: An improved convection scheme applied to recombining divertor plasma flows. *J. Comput. Phys.,* **142,** 473–488.

——, P. McHugh, and D. Keyes, 1996: Newton–Krylov methods for low-Mach-number compressible combustion. *AIAA J.,* **34,** 961–967.

——, D. B. Kothe, and B. Lally, 1999a: A new nonlinear solution method for phase-change problems. *Numer. Heat Transfer,* **35B,** 439–459.

——, W. J. Rider, and G. L. Olson, 1999b: An efficient nonlinear solution method for non-equilibrium radiation diffusion. *J. Quant. Spectrosc. Radiat. Transfer,* **63,** 15–29.

McHugh, P. R., and D. A. Knoll, 1994: Fully implicit finite volume solutions of the incompressible Navier–Stokes and energy equations using inexact Newton's method. *Int. J. Numer. Methods Fluids,* **18,** 439–455.

Mousseau, V. A., D. A. Knoll, and W. J. Rider, 2000: Physics-based preconditioning and the Newton–Krylov method for non-equilibrium radiation diffusion. *J. Comput. Phys.,* **160,** 743–765.

Nash, S. G., 1985: Precondition of truncated-Newton methods. *SIAM J. Sci. Stat. Comput.,* **6,** 599–616.

——, and A. Sofer, 1990: Assessing a search direction within a Truncated-Newton method. *Oper. Res. Lett.,* **9,** 219–221.

Pernice, M., and H. F. Walker, 1998: NITSOL: A Newton Iterative Solver for Nonlinear Systems. *SIAM J. Sci. Comput.,* **19,** 302–318.

Pielke, R. A., and Coauthors, 1992: A comprehensive meteorological modeling system—RAMS. *Meteor. Atmos. Phys.,* **49,** 69–91.

Qin, N., D. K. Ludlow, and S. T. Shaw, 2000: A matrix-free preconditioned Newton/GMRES method for unsteady Navier–Stokes solution. *Int. J. Numer. Methods Fluids,* **33,** 223–248.

Reisner, J. M., S. Wynne, L. G. Margolin, and R. R. Linn, 2000: Coupled atmospheric–fire modeling employing the method of averages. *Mon. Wea. Rev.,* **128,** 3683–3691.

Saad, Y., 1995: *Iterative Methods for Sparse Linear Systems.* International Thompson Publishing, 447 pp.

Schär, C., and P. K. Smolarkiewicz, 1996: A synchronous and iterative flux-correction formalism for coupled transport equations. *J. Comput. Phys.,* **128,** 101–120.

Skamarock, W. C., and J. B. Klemp, 1992: The stability of time-split numerical methods for the hydrostatic and nonhydrostatic elastic equations. *Mon. Wea. Rev.,* **120,** 2109–2127.

——, P. K. Smolarkiewicz, and J. B. Klemp, 1997: Preconditioned conjugate-residual solvers for Helmholtz equations in nonhydrostatic models. *Mon. Wea. Rev.,* **125,** 587–599.

Smolarkiewicz, P. K., 1984: A fully multidimensional positive definite advection transport algorithm with small implicit diffusion. *J. Comput. Phys.,* **54,** 325–362.

——, and L. G. Margolin, 1993: On forward-in-time differencing for fluids: Extension to a curvilinear framework. *Mon. Wea. Rev.,* **121,** 1847–1859.

——, and ——, 1994: Variational solver for elliptic problems in atmospheric flows. *Appl. Math. Comput. Sci.,* **4,** 527–551.

——, and ——, 1997: On forward-in-time differencing for fluids: An Eulerian/semi-Lagrangian nonhydrostatic model for stratified flows. *Atmos.–Ocean,* **XXXV,** 127–152.